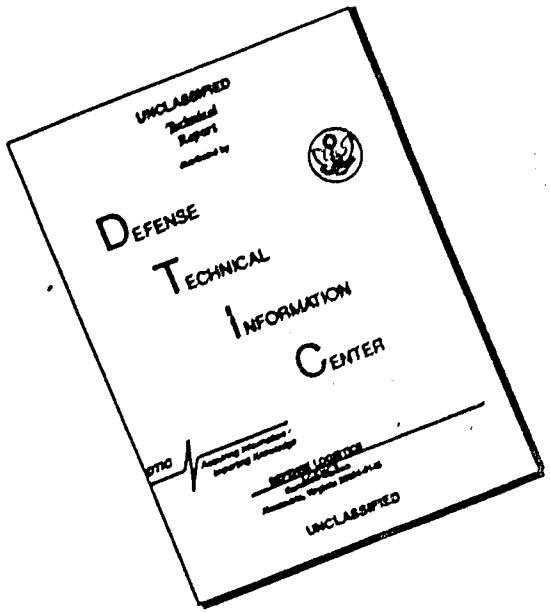
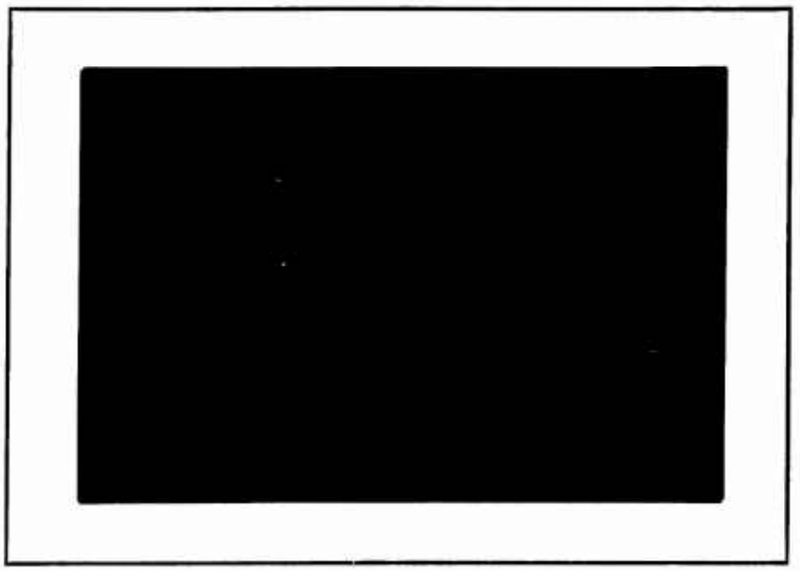
DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

64-11

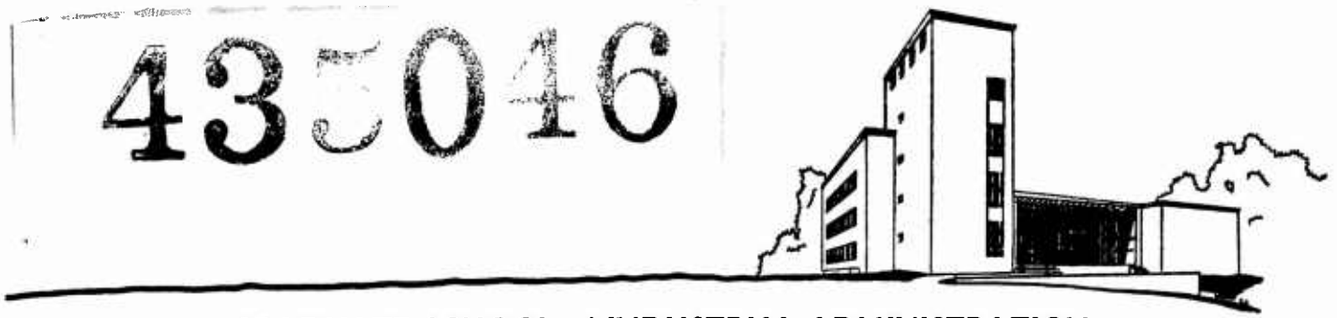
(1)



Carnegie Institute of Technology

Pittsburgh 13, Pennsylvania

435046



GRADUATE SCHOOL of INDUSTRIAL ADMINISTRATION

William Larimer Mellon, Founder

(14) [illegible] ONR-[illegible]116

~~O.N.R. Research Memorandum No. 116~~

(6) A STUDY OF THE ALL-INTEGER

INTEGER PROGRAMMING ALGORITHM*

(10) by

Fred Glover

(11) September 1963

(1) 16 [illegible]

Graduate School of Industrial Administration

Carnegie Institute of Technology

Pittsburgh 13, Pennsylvania

*The author would like to acknowledge his indebtedness to Professor G. L. Thompson for several valuable suggestions in the preparation of this paper.

This paper was written as part of the contract, "Planning and Control of Industrial Operations," with the Office of Naval Research, at the Graduate School of Industrial Administration, Carnegie Institute of Technology. Reproduction of this paper in whole or in part is permitted for any purpose of the United States Government. Contract ~~O.N.R. 99~~.

(1) [illegible] N-[illegible] 23000

AI

1. INTRODUCTION. Since its inception integer linear programming has, paradoxically, been a source of both promise and disappointment. Promise because there are manifold and compelling opportunities for its application; disappointment because it has made only the most dubious progress in spite of these opportunities.

In contrast, the remarkable growth of traditional linear programming is well known. The question arises as to what causes this difference. Harold Kuhn and Richard Quandt cite three factors to account for the success of linear programming. 1/ They are (a) its flexibility and wide range of application, (b) the parallel development of large electronic computers, and (c) the discovery of the simplex method, an efficient algorithm which solves problems at a reasonable cost in time. Of these factors, the one which seems most nearly to explain the contrast between the ascension of traditional linear programming and the lagging debut of its integer relative is the last. There are no integer programming algorithms with the high efficiency of the simplex method.

The difficulties with the integer algorithms are of two distinct types. First, there are many problems which appear wholly impervious to them, though the algorithms are theoretically assured of finding a solution. Second, the algorithms give fickle performances even in solving those problems which are more tractable. For this latter category of problems the success of the integer algorithms depends heavily on the sequence of selecting pivotal constraints. Frequently a problem which requires an inordinate number of pivots using one sequence will yield after a fraction of that number to another.

It is sometimes possible, by examining an efficient pivot sequence for a specific problem, to infer a general rule which will work well on the

problem investigated. Unfortunately, there seems to be no existing rule which is good for all problems, nor which makes any appreciable inroad into the large region of problems which uniformly resist the efforts of the best alternate approaches.

The initial purpose of this study was to seek a strategy of pivot selection that would abandon the profligacy of existing rules. The algorithm at the focus of our investigations was to be, and has principally been, the all-integer integer programming algorithm proposed by Ralph Gomory in January 1960. 2/ However, as our work has progressed, we have developed two supplementary techniques, based on the simplex method, which may be applied independently of any specific rule of choice. One of these, the New Origin Technique (NOT), described in Section 4B, has proved sufficiently promising to earn a distinguished position among the strategies presented in this report. Thus we have come to attack the tardiness of the integer programming algorithm in two ways: by developing and testing rules for selecting pivotal constraints, and by introducing two methods for boosting the problem toward solution before the pivot rules are applied.

2. THE ALL INTEGER INTEGER PROGRAMMING ALGORITHM. Gomory's all integer integer programming algorithm resembles the simplex in several ways. The distinguishing feature of the algorithm is the creation of a transitory constraint which is used for pivoting and then discarded. Underlying this feature is the principle that each constraint in the all integer tableau implies the existence of a special set of secondary constraints. The secondary constraints must all be satisfied by any integer solution to the original problem, and are eligible for pivoting if and only if the constraint which sired them is eligible. The process of pivot selection consists in choosing one of the eligible constraints from the tableau, and then locating the

unique secondary constraint which satisfies the following two properties: (1) it has a pivot element of -1, and (2) of those constraints sharing property one, it causes the largest decrease in the objective function. The "new pivotal constraint" is temporarily annexed to the tableau so that the pivot operation may be carried out on it. Afterwards it is discarded, and the process repeated.

In Gomory's tableau the constraints correspond to column vectors. The new pivotal constraint is uniquely determined by the eligible constraint from which it is derived. Hence we will speak of pivoting on a column in the permanent tableau, though the actual pivot operation is carried out on the derivative column. For the convenience of the reader, Gomory's algorithm is outlined in its entirety in Appendix A.

3. PIVOT SELECTION RULES. A column in Gomory's tableau is eligible for pivoting whenever its bottom row element is negative, hence the choice of pivots may be referred to as a choice among negative indicators. There are clearly many strategies by which such choices could be made. Though some limited value might come from testing rules that have been fabricated arbitrarily, we have chosen to restrict ourselves to rules for which we could give some intuitive, practical or theoretical justification.

A. First Negative Indicator.--The first negative indicator rule says to select the eligible column with the least index. In his paper on all integer programming Gomory cites this rule as one for which a finite number of pivots is guaranteed. Its advantage is its extreme simplicity; the computer wastes no time between pivot operations. Compared to any rule which requires the same number of pivots, the first negative indicator rule must prove at least as efficient.

B. Random Pivot Selection.--The name of this rule is its description; it simply directs that a negative indicator be chosen at random. Like the

first negative indicator rule, the random rule works rapidly. Moreover, it is a normative rule: hopefully we should expect any truly effective rule to do as well. Finally, the random rule makes it possible to get an idea of the distribution of the number of pivots required to reach solution by making successive solution attempts on a single problem with a variety of random number bases.

C. Random Restart Rule.--Our early experimentation with the random pivot choice rule showed us that the variation in the number of pivots required to solve even the more manageable problems by Gomory's algorithm was exceedingly large. Small problems were almost as notorious in this respect as larger ones. A six variable six inequality problem had to be taken off the computer unsolved after 5,600 pivots using one random number base, yet was solved in less than 40 pivots using another. The thought naturally arises that a program for arresting work on a problem after a pre-established number of pivots, and then starting anew, might pay off in more consistent results. Any stop rule at these early stages of experimentation must be rather tentative and arbitrary, and part of our efforts in this area have accordingly been to find a guide by which such a rule could be set.

D. Random Boost Rule.--The random boost rule is an attempt to make use of information which the Random Restart Rule passes by. Each try for a solution with Gomory's algorithm establishes a lower bound to the objective function. The random boost rule embodies this lower bound in a new constraint which is pivoted on and satisfied before all others are considered. Once the objective function attains the lower bound of the previous solution attempt, the algorithm proceeds for a specified number of pivots and then passes along a possibly new bound to the next attempt.

There is considerable similarity between this rule and the New Constraint Technique in Section 4A. For an elaboration of the mechanics of the procedure see the discussion of the latter

E. Maximum Frequency Eligible Rule.--The Maximum Frequency Eligible Rule is to select that constraint for pivot which was most frequently eligible in the past. Its use was suggested by experience in hand testing Gomory's algorithm, where it was found that an excessive number of pivots was often required to reach a solution when avoiding a constraint consistently eligible.

F. Cardinal Learning Pivot Rule.--This rule and the next, the ordinal learning pivot rule, take a new direction in pivot choice strategies. They are based on the psychological model of reinforcement learning, according to which the organism can be induced to acquire a systematic pattern of behavior by rewarding certain responses and punishing others. A formal structure of multiple-choice situations has been developed[3] which has already been applied outside psychology by Fischer and Thompson in the construction of job shop scheduling rules.[4] The present attempt is to extend the model to the process of pivot selection in integer programming.

The mathematical model of reinforcement learning posits that the organism is endowed with a probability vector describing the likelihood with which he will select each of a set of alternatives put before him. The choice of an alternative which meets with reward produces a change in the probability vector which increases the chances that the alternative will be chosen again. Similarly, a punished alternative results in a decreased likelihood that it will be chosen again.

The cardinal learning pivot rule is designed in this fashion to reward and punish good and bad pivot choices. A "good" pivot choice is defined to be one which produces a relatively large increase in the objective function; a "bad" choice, similarly, implies a "small" increase. In the routine we have used, a performance record is kept for each column in the tableau, measuring the mean contribution to the objective function for which that column was responsible. (Other criteria of performance are of course possible; for instance the maximum objective function change could be used instead of the average, or

the fractional change substituted for the absolute. At a more refined level, the probability weights assigned to the choice of columns might be based on the trend of their performance, higher priority going to a column whose performance is steadily improving than to one whose performance is deteriorating. An additional refinement would be to consider columns in conjuntion, gauging the relative merits of alternative cycles of pivot choice.)

The particular version we programmed for the computer initially segregates the eligible columns into three groups: (a) the columns which have been chosen for pivoting in the past, (b) those columns which are presently eligible for pivoting, and also members of the first group, (c) those columns which are presently eligible for pivoting, but which are not members of the first group.

Associated with each column in group (a) is the information telling the total objective function change due to pivoting with that column, and the number of times which the column was used for pivoting. From this the average objective function change due to choosing that column for pivot is computed, and then squared. We use the squares rather than the means themselves in determining the probability with which each eligible column is to be selected in order to accentuate the difference in weights assigned to columns with good and bad performance histories. The probability weight assigned to a member of group (b) is simply the squared value for which it is responsible. Each member of group (c), on the other hand, receives the same probability weight, which is equal to the mean of the squares associated with group (a).

This rule is called the cardinal rule because each column which has previously been pivoted on is always assigned a weight determined by its individual history; for example, column number 7 will be considered for pivoting according to its own past performance, rather than on the fact that, from left to right in the tableau, it is at present the third of the columns which are eligible

To represent the rule precisely, let a, b, and c denote the number of columns in (a), (b), and (c), respectively. The term m_i will denote the mean objective function change for which column i is responsible, and the symbol X will be used to denote the set of all subscripts i for which column i is in (x). Then the probability p_k to be assigned to selecting the eligible column k for pivot may be represented as follows.

I. If $k \in B$:

$$p_k = m_k^2 / \left(\sum_{i \in B} m_i^2 + (c/a) \sum_{j \in A} m_j^2 \right)$$

II. If $k \in C$:

$p_k = \left(\sum_{j \in A} m_j^2 \right) / w$, where w is the denominator in I above.

G. Ordinal Learning Pivot Rule.--This rule, like the cardinal rule, attempts to encourage good choices with reward, and discourage bad choices with punishment (or "negative" reward). The vehicle of reward, positive and negative, is a successively redefined vector which expresses the probability with which each column will be chosen for pivot at any particular stage of the algorithm. Eligibility acts as a restrictive operator which automatically assigns zero probability to those columns which do not have negative indicators, and calls for the adjustment of the other probabilities accordingly. The ordinal rule differs from the cardinal in that the probability assigned to selecting a particular column for pivot is not dependent solely upon its own history, but upon its ordinal position among the eligible columns in the tableau. That is, we will be more interested in the fact that a column is the third from the left of those which are eligible, rather than its precise index in the tableau.

As it is true of the cardinal rule, there are a number of ways in which the ordinal rule might be formulated. An intuitive version would be to establish an "ordinal probability weight vector" containing a number of components equal to some multiple of the number of columns, and keep an "ordinal history"

for that vector independent of the performance records of the individual columns.

As an illustration, if the tableau had twelve columns, the ordinal vector sixty components, and five columns were eligible for pivoting, then the probability weight to be attached to the eligible column with the least index would be equal to the sum of the first twelve components of the ordinal vector, the probability weight for the column with the next highest index the sum of the next twelve components, and so on.

When the pivot was executed, the result would be recorded by maintaining two supplementary vectors, one to keep track of the objective function changes, and the other to keep track of the number of pivots. If the third of the five eligible columns was selected, the objective function change would be added to each of the amounts in columns 25 through 36 in the first supplementary vector, and in the second supplementary vector the same columns would be incremented by one. In this way a form of average objective function change could be computed to provide the probability weights to be inserted in the ordinal probability weight vector.

The version which we used might be called a hybrid ordinal rule in the sense that the supplementary vectors are designed to associate each column with its own history. The ordinal weight vector itself, however, works as in the preceding illustration. For example, again suppose there were twelve columns in the tableau, five of which were eligible. In this case the ordinal vector would have just twelve components, the same as the number of columns in the tableau. Each component would be equal to the (squared) mean objective function change due to pivoting on the corresponding column. columns not previously pivoted on treated the same as in the cardinal learning routine. Then the probability weight to be assigned to the first eligible column would be equal to the sum of the first two and two fifths components, the probability weight for the second eligible column equal to the sum of the next two and two fifths components, etc.

To represent the rule precisely, we use the notation used to describe the cardinal rule. In addition we define (d) to be the set of all columns, (e) to be the set (d) - (a), and (f) to be the set union of (b) and (c). We let $v_i = m_i^2$ if $i \notin B$, and $v_i = (\sum_{j \in B} m_j^2)/a$ if $i \in B$. Let $j^* = \min(j: j \notin F)$. Then let $'$ be defined: if $j \in F - j^*$, then $j' = k \notin F$, such that $k \lneqq j$, and if $i \lneqq j$, $i \in F$, then $i \leqq k$. Finally, define $q_{j^*} = \underline{\text{int}}\ (d/(b+c))$, $r_{j^*} = d/(b+c) - q_{j^*}$. Then the probability p_k to be assigned to choosing the eligible column k for pivot may be represented:

I. If $k = j^*$:

$$p_k = \sum_{i=1}^{q_{j^*}} v_i + r_{j^*} v_{q_{j^*}+1}$$

II. If $k \neq j^*$:

$$p_k = \left((1 - r_{k'}) \cdot (v_{q_{k'}+1}) + \cdot \sum_{i=q_{k'}+2}^{q_k} v_i + r_k v_{q_k+1}\right) / \sum_{j \in E} v_j$$

where $r_k = q_{j^*} + r_{j^*} + r_{k'} - 1 - g_k$, $q_k = g_k + q_{k'} + 1$

and $g_k = \underline{\text{int}}\ (q_{j^*} + r_{j^*} + r_{k'} - 1)$

H. Largest Objective Function Change.—The name of this rule is largely self-explanatory. We learned while working with problems simple enough to be solved by hand that we often got close to the minimum number of pivots by choosing the columns which individually produced the largest objective function changes. In consequence we decided to test the rule on the computer with more complex problems.

Because of the fairly extensive computation needed to apply this rule, it must result in a fairly small number of pivots to compare favorably with, say, either the least index rule or the random choice rule. In fact, in the order we've presented them, the pivot rules begin with the one requiring the least amount of computer time on any given pivot and extend to the one requiring the most. If all are to compare equally well in respect to the mean time

required to solve problems, the later rules must result on the average in fewer pivots than the earlier ones.

A SUPPLEMENTARY TECHNIQUE. The supplementary techniques were developed in an attempt to get a "head start" with the Simplex before applying Gomory's all integer algorithm. We felt that there were two reasons that might make such an attempt worthwhile. (1) As problems get more difficult, the number of pivots required by the simplex method compare more and more favorably with the number required by the all integer algorithm; (2) the boost from the simplex method could possibly simplify the problem so that expedient rules of choice would subsequently work as well as more sophisticated ones, cutting down the time consumed in each pivot operation.

A New Constraint Technique (NCT). -- One of the most conspicuous and most easily used pieces of information given by the simplex algorithm is the value of the objective function. Whatever this value is for the fractional solution, we know that the value for the integer solution (in the minimum problem) must be greater than or equal to it. The NCT, like the Random Boost Rule, involves the creation of a new constraint expressing this inequality, and which is then annexed to the regular integer programming tableau. The initial pivots with Gomory's algorithm are carried out on the new constraint until it is satisfied, after which it may be discarded (it can never be violated) and the problem reverts to its usual form. The motivation for this approach was the expectation that (1) the number of pivots required to satisfy the new constraint would be few, and (2) advancing the objective function closer to the solution vicinity may result in fewer pivots being needed thereafter.

The NCT can be represented formally as follows. We refer to the standard form of the minimization problem: "minimize $wb + b_0$," subject to $wA \geq c$ and $w \geq 0$." The new constraint is given by $wb \geq b_0 + \langle w' b \rangle$ where w' is the Simplex solution vector.

One apparent advantage of the NOT, aside from its simplicity, is that the effort put into the simplex algorithm is never "lost"---no matter what the form of the problem, there is always sufficient information to form the new constraint.

B. The New Origin Technique (NOT).--The NOT is the most effective of the strategies in this report. It has been able to solve 62% more eight variable eight inequality problems within a 400 pivot cutoff limit than the second best rule tested, and in general requires from 40% to 60% fewer pivots to solve those problems which the other methods were able to solve within the cutoff limit. The NOT uses the simplex algorithm, and in particular information obtained from a tableau produced at some stage of the algorithm, to give starting values to the variables before applying Gomory's algorithm. Implicitly, Gomory's algorithm normally takes the starting values for all the w_i to be zero. To express the condition that some of the variables begin at other than a zero value, we state the following lemma.

Lemma 1. Given any vector w^0, if w^1 belongs to the solution set of the problem "minimize $wb + b_0 + w^0b$, subject to $wA \geq c - w^0A$ and $w \geq -w^0$," then there is a w^* in the solution set of the standard problem "minimize $wb + b_0$, subject to $wA \geq c$ and $w \geq 0$," so that $w^* = w^1 + w^0$, and conversely.

From the hypothesis, $w^1A \geq c - w^0A$ and $w^1 \geq -w^0$, hence $(w^1 + w^0)A \geq c$ and $(w^1 + w^0) \geq 0$, so that $w^1 + w^0$ is a feasible solution of the standard problem. Since w^1 minimizes $wb + b_0 + w^0b = (w + w^0)b + b_0$, the vector $(w^1 + w^0)$ also obviously minimizes the function $wb + b_0$. The converse follows by considering the vector $(w^* - w^0)$.

Customarily, of course, w^0 is the zero vector. In what follows we will attempt to compute a w^0 some of whose components are possibly other than zero. We will refer to this computed w^0 as the new origin. The justification for the NOT is the assumption that a good choice of an initial w^0 would reduce the number of pivots required to reach a solution with Gomory's algorithm. In the

extreme case, if w^0 were an integer solution, no pivots would be required. However, a bad choice of w^0 would mean that the correct answer could not be obtained.

Two impositions must be placed upon the new origin if a correct solution is to be obtained with the all integer algorithm. We embody these in

Lemma 2. To insure that Gomory's algorithm will obtain a solution w^1 (to the first problem in Lemma 1) so that $(w^1 + w^0)$ minimizes the standard problem (i) the components of w^0 must all be integer, and (ii) each component of w^0 must be less than or equal to the corresponding component of w^*.

If (i) were not true, the constraints $w \geq -w^0$ would not be permissible in the all integer tableau. We note that these constraints in the new problem are not to be coupled with the constraints $w \geq 0$, but replace them.

To see the validity of (ii), we observe that in transforming the standard problem into one with the form "minimize $wb + b_0 + w^0b$, subject to $wA \geq c - w^0A$, $w' = w^0$," the same result is achieved as by first adding the constraints $w_i \geq w_i^0$ to the standard problem, pivoting on them before all others, and then ignoring them. Hence if it is false in the solution to the standard problem that $w_i^* \geq w_i^0$ for some i, then the solution obtained by Gomory's algorithm will be wrong unless the ignored constraint is violated in the proper degree during the course of pivoting.

Using information obtained from the simplex solution of the linear programming problem, a new origin vector can be derived from any one of the simplex tableaus. In order to describe the ensuing derivations straightforwardly, it is convenient at this point to refer to a form of the simplex algorithm which solves the minimization problem directly rather than as the dual of the maximum.

We will call the standard representation of the problem--minimize $wb + b_0$ subject to $wA \geq c$ and $w \geq 0$, --Formulation I. The initial tableau for the version of the simplex algorithm which solves this standard minimization problem directly is shown in (a) below.

Those columns in the tableau (a) which are eligible for pivoting are negative in the -c vector. The pivot element is chosen from among the negative entries in the selected column so that the rows of the tableau will remain lexicographically positive after the pivot.

For additional simplicity we denote the tableau matrix consisting of -A and the identiy matrix I by the single letter A, replace b and 0 vector by b alone, replace -c by c, and extend the w vector so that it includes the slack variables. The problem can then be rewirtten: minimize $wb + b_0$ subject to $wA = c$, $w \geq 0$, as represented by the tableau (b). We shall henceforth refer to this as Formulation II.

The simplex algorithm is a recursive transformation which changes each statement of the problem into an equivalent statement, summarized by the subsequent tableau, with the same solution set as the original. Hence Formulation II is perfectly general, and (b) may represent any of the simplex tableaus.

(a)

b	-A
0	I
$-b_0$	-c

(b)

b	A
$-b_0$	c

We will abide by the convention that b_0' denotes the value of b_0 in the final simplex tableau. Since b_0' is the (possibly) fractional solution value of the function $wb + b_0$ found by the simplex algorith, the integer variable solution value of $wb + b_0$ must be greater than or equal to b_0' rounded upward. Using the notation $\langle x \rangle$ to denote the smallest integer greater than or equal to x, we may state:

Lemma 3. The constraining equation

(1) $wb = \langle b_0' \rangle \equiv b_0 + p$

is true for some integer $p \geq 0$ for the integer problem, where b and b_0 refer to the values from any simplex tableau, as indicated in Formulation II. (The lemma must also be true, of course, of Formulation I.)

The proof follows from the foregoing remarks, since without requiring that the variables be integral, $w'b = b_0' - b_0$, where w' is the simplex solution vector.

We now proceed to the main derivation of the new origin vector w^0.

By Formulation II, each constraint from a given simplex tableau can be written

(2) $\sum a_k w_k = c$.

Let i be defined by $a_i/b_i = \min (a_k/b_k)$ for all k such that not both a_k and b_k are zero. By convention $a_k/0 = -\infty$ if $a_k < 0$ and $a_k/0 = \infty$ if $a_k > 0$. It is assumed that i is unique.

Let $H = \min (a_k/b_k)$ for all $k \neq i$, excluding $a_k/b_k = 0/0$. We immediately state the following lemma.

Lemma 4. Provided $H \neq \infty$, (i) $Hb_k - a_k \leq 0$ if $k \neq i$: (ii) $Hb_i - a_i \geq 0$.

We note that part (i) holds trivially if a_k and b_k are both zero. If we assume otherwise, the first part then follows from the fact that $H \leq a_k/b_k$ if $k \neq i$. By hypothesis $H \neq \infty$, and since i is required to be unique, $H \neq -\infty$. Thus $b_k \neq 0$ for all k such that $a_k \neq 0$. But by the simplex algorithm, $b_k \geq 0$ for all k, hence $b_k > 0$ for all relevant k, $Hb_k \leq (a_k/b_k)\, b_k = a_k$, and $Hb_k - a_k \leq 0$. Similarly, $Hb_i - a_i > 0$ follows from the fact that $a_i/b_i < H$.

We will use Lemma 4 to derive a permissible value for the component w_i^0 of the new origin vector w^0. Rewriting equation (1) as a sum of variables and multiplying both sides we obtain

(3) $\sum Hw_k b_k = HB$, where B is defined by $B = \langle b_0^* \rangle - b_0 + p$.

Subtracting (2) from (3) gives

$$\sum w_k(Hb_k - a_k) = HB - c$$

We transpose for $k \neq i$, and invoke Lemma 4 to obtain

$$(Hb_i - a_i)\, w_i = HB - c - \sum_{k \neq i} w_k (Hb_k - a_k) \geq HB - c.$$

Again, since $Hb_i - a_i > 0$,

(4) $w_i \geq (HB - c)/(Hb_i - a_i)$

from which we can infer a permissible value for w_i^0. If we further define j by $-a_j/b_j = \min (-a_k/b_k)$ for all k, and $Q = \min (-a_k/b_k)$ for all $k \neq j$, (again excluding $a_k/b_k = 0/0$) we can state the following theorem, the first part of which we have just proved.

Theorem 1. Whenever the following expressions are well-defined and greater than or equal to zero, permissible values for the components w_i^0 and w_j^0 of the new origin vector w^0 are given by

(i) $w_i^0 = \langle (HB - c)/(Hb_i - a_i) \rangle$

(ii) $w_i^0 = \langle B/b_i \rangle$ if $H = \infty$

(iii) $w_j^0 = \langle Qb + c \rangle$

Part (ii) of the theorem follows from the fact that if $H = \infty$, then $b_k = 0$ for all $k \neq i$. Hence equation (1) reduces to $w_i b_i = \langle b_0^* \rangle - b_0 + p = B$, and for the strictly correct value for p, $w_i = B/b_i$. We write this in the form (ii) since, as we will show later, it may suffice to give p a value slightly lower than the one required to make equation (1) an identity for the all integer solution vector $w = W^*$. The case for $b_i = 0$ is omitted since $b_k = 0$ for all k, which makes the objective "minimize $wb + b_0$" meaningless. For this reason $Q = \infty$ is also excluded from consideration. The derivation of (iii) begins with $\sum (-a_k)w_k = -c$ in place of equation (2). The remainder of the derivation is a duplication of that of (1), with Q, j, $-a_k$, and $-c$ substituted for H, i, a_k, and c. Thus we get $w_j \geq (Qb + c)/(Qb_j + a_j)$ as the

complement of (A_j). However, from the knowledge that each constraint contains a basic variable, we can conclude that $\min(-a_k/b_k) = -1/0$, and $-b_j + a_j = 1$. Part (iii) follows and the proof of Theorem 1 is complete. We remark that any component of the new origin vector w^0 which is left undetermined by the equations of Theorem 1 is automatically assigned the value zero. If a component of w^0 is determined to have different non-zero values by several constraints, it is reasonable to assign it the largest of the values.

For a given constraint, it is not always necessary to compute both w_i^0 and w_j^0, as the following illustrates.

Theorem 2. Provided $H \neq \infty$, (i) if $w_i^0 > 0$, then $w_j^0 = 0$, and (ii) if $w_j^0 > 0$, $w_i^0 = 0$. To prove this we first prove the following two lemmas.

Lemma 5. $B \geq 0$.

Lemma 6. If $H \neq \infty$, then $Q \leq -1$.

Lemma 5 follows by recalling the definition $B = \langle b_0' \rangle - b_0 + p$, $p \geq 0$ (p integer), and noting that the simplex algorithm increases the value of b_0 monotonically, so that $b_0' \geq b_0$.

In Lemma 6, $-H = a_m/b_m$ for some subscript m other than i. Since $-H = \min(-a_k/b_k)$ for all $k \neq j$, $Q \leq -H$ unless $m = j$. But then $H = a_j/b_j = 1/0 = \infty$.

To prove (i) of Theorem 2, we note that, since $Hb_i - a_i > 0$, if $W_i^0 = \langle (HB - c)/(Hb_i - a_i) \rangle \geq 0$, then $HB - c > 0$. Thus $-HB + c < 0$, and since $-H \leq -H$ and $B \geq 0$, $-B - a \leq -HB + c < 0$, from which $w_j^0 = 0$ follows at once. The proof of (ii) proceeds similarly. It may parenthetically be remarked that the provision $H \neq \infty$ in Theorem 2 is more restrictive than necessary. Since $H = a_m/b_m = \min(-a_k/b_k)$ for $k \neq i$, $H = \infty$ implies $a_k/b_k = \infty$ for all k other than $k = i$ and $k = m$, provided not both a_k and b_k equal zero. Hence if there is an index n for which $a_n \neq 0$, $n \neq m$, $n \neq i$, then $a_n/b_n = a_m/b_m = \infty$, and the subscript j defined by $-a_j/b_j = \min(-a_k/b_k)$ for all k, is nonunique, hence $w_j^0 = 0$ by default.

The last remark defines the limit to which Theorem 2 may be strengthened. It is not true, for example, that at least one of w_i^0 or w_j^0 must be positive. If zero is the only permissible value for w_i^0 there is no guarantee that w_j^0 will be any better.

In order to use Theorem 1 to derive a new origin vector w^0, it is necessary to find an acceptable value for p. We note that (1) must be true for the integer solution vector $w = w^*$, and we obtain the defining equation

(5) $p^* = w^*b + b_0 - \langle b_0' \rangle$

If b and b_0 are taken from the initial tableau, it can be seen that p^* represents the difference between the objective function value for the all integer solution ($w^*b + b_0$) and the objective function value for the solution with the simplex algorithm ($b_0' = w'b + b_0$, w' the simplex solution vector), where the latter has been rounded to its next highest integer. While p^* is the one strictly correct value for p, if $H < 0$ for all the constraints for which $w_i^0 > 0$, the only danger is in choosing p too small, since only then would the w_i^0 be larger than implied by the correct value. The same is true of Q and the w_j^0. Conversely, for H or $Q > 0$, the risk is in choosing p too large.

It is fortunately possible, by being systematic, to reduce both cases to one. Gomory's algorithm, like the simplex algorithm, produces a new objective function value in place of the original b_0 with each successive pivot. For any specified tableau obtained by Gomory's algorithm we will denote the current objective function value by B_0.

The procedure is to begin with $p = 0$, determine the w_i^0 and w_j^0 and start Gomory's algorithm. There will be some cutoff value for B_0 which, if transcended, will indicate that the choice of p was unsafe and that the w_i^0 and w_j^0 need to be revised. At this point p is incremented by an appropriate amount, the new w_i^0 and w_j^0 determined, and the process repeated. The process can be formulated precisely as follows.

The NOT Procedure:

(a) Let $p_0 = 0$

(b) Determine the new origin vector w^0 on the basis of $p = p_0$.

(c) Determine the least integer $p_1 > p_0$ such that assigning p the value p_1 will yield a different value for some component of w^0 than by giving p the value p_0.

(d) Apply Gomory's algorithm to the problem "minimize $wb + b_0 - w^0 b$ subject to $wA \geq c - w^0 A$, $w \geq -w^0$," where b, b_0, A, and c are from the initital formulation (see Formulation I, page 13).

(e) Let B_0 represent the current objective function value taken from a specified tableau for the problem in (d). If, after any pivot, $B_0 > \langle b_0 \rangle + p_1$, give p_0 the value p_1 and begin again at (b).

(f) If the condition in (e) is not satisfied, w^0 was suitably chosen. The objective function value $B_0 = b_0^*$ and the values of the variables w_j displayed in the final tableau to the problem in (d) are the solution values for the standard problem "minimize $wb + b_0$ subject to $wA \geq c$, $w \geq 0$."

We remark first on the second statement in (f). Let w^1 denote the all integer solution vector to the problem in (d). Under the assumption that w^0 is suitably chose, we have shown that $w^1 + w^0$ is the all integer solution vector to the standard problem in (f). The objective function value $B_0 = b_0^*$ given by the final Gomory tableau for the problem in (d) satisfies the equation $b_0^* = w^1 b + b_0 + w^0 b = (w^1 + w^0)b + b_0$ and hence is the solution value for the objective function to the problem in (f). The effect of the constraints $w \geq -w^0$ in the initial Gomory tableau is to replace the zero vector beneath the -I matrix for the standard problem with the vector w^0. This "starts" each w_j at w_j^0 rather than at 0, so that the final w vector read from the tableau is $(w^1 + w^0)$ rather than w^1, as the second part of (f) claims

We will now prove the validity of the NOT procedure with the following theorem.

Theorem 3. Assume that the NOT procedure has been applied as specified above.
(i) If $B_0 > \langle b_0' \rangle + p_1$, then the strictly correct value of p, $p^* \geq p_1$.
(ii) If $b_0^* \leq \langle b_0' \rangle + p_1$, then $p_0 \leq p^* \leq p_1$, and b_0^* is the correct objective function solution value.

We first prove Theorem 3 under the hypothesis $p_0 \leq p^*$.

To prove (i), suppose $p^* < p_1$. By the defining equation (5) of p^*, $w^*b + b_0 - \langle b_0' \rangle < p_1$, and hence $w^*b + b_0 < \langle b_0' \rangle + p_1$. Applying the hypothesis of (i) to this last equation we get

(6) $w^*b + b_0 < B_0$.

But if $p_0 \leq p^* < p_1$, then the choice of p_0 determines the same w^0 vector as p^*. Therefore, as we have remarked, Gomory's algorithm applied to the problem in (d) of the NOT procedure must display the correct solution to the problem in (f) in its final tableau. Since the algorithm increases the current objective function B_0 monotonically with each pivot, $B_0 \leq w^*b + b_0$, contradicting (6). Consequently $p^* \geq p_1$.

For (ii) we suppose $p^* > p_1$. Then from (5), $w^*b + b_0 - \langle b_0' \rangle > p_1$ and $w^*b + b_0 > \langle b_0' \rangle + p_1$. By the hypothesis of (ii) this yields $w^*b + b_0 > b_0^*$, or in other words the integer objective function value b_0^* associated with a feasible solution $(w + w^0)$ to the problem in (f) is smaller than the minimum objective function value $w^*b + b_0$, which is impossible. The conclusion $p^* \leq p_1$ follows.

To prove the second half of (ii) we note that if $p_0 \leq p^* < p_1$, then p_0^* determines the same w^0 vector as p^* and b_0^* is the correct objective function value of (f). On the other hand, if $p^* = p_1$, then by the same sequence of reasoning as in the preceding paragraph $b_0^* \leq w^*b + b_0$, and hence $b_0^* = w^*b + b_0$, confirming that b_0^* is still the correct solution, and (ii) is true.

The complete proof follows the preceding pattern inductively by starting as in the NOT procedure with $p_0 = 0$. For this the hypothesis $p^* \geq p_0$ is satisfied, and the proof of (i) shows that $p^* \geq p_0$ must continue to hold as long as the NOT procedure is iterated according to instruction (e).

The validity of the NOT procedure follows immediately from Theorem 3 since by (i) is is assured that the condition in (e) will not be satisfied unless the procedure can subsequently find a value for p_0 which will produce the same w^0 vector as p^*, and by (ii), it is assured that the procedure will stop with b_0^* equal to the correct objective function value.

The question arises as to which simplex tableau gives the best new origin. The final one seems a logical choice since in the immediate vicinity of the simplex solution the only variables which must have positive values will be in the simplex basis. As p grows larger, however, the possibility of new variables becoming implicitly nonzero will increase. In the actual tests made with the NOT, both the initial and final tableau were used to generate the w_i^0 and w_j^0 and the values compared.

It might be guessed that since each tableau implies the same solution set, the values for the w_i^0 and w_j^0 would be the same regardless of which tableau was selected for their derivation. Interestingly enough, this supposition is wrong. It is even possible to construct problems for which the values of some of the components of w^0 are worse from a later tableau than from an earlier one. A uniformity that does hold, however, is that for the constraint used as pivot the i and j subscripts of w_i^0 and w_j^0 switch values, and the new w_j^0 equals the old w_i^0, and vice versa. The proof of this is given in Appendix B.

There is additionally the question of whether the NOT ought to be used to assign positive values to the slack variables. The argument against assigning values to the slacks is that this increases the requirements to be met by the nonslacks. In consequence it becomes more likely that too large a value for the objective function will be implied. On the other hand, positive slack

values may speed up Gomory's algorithm if they are nearly correct. The question here ties in closely with the question of the new origin technique and feasibility, soon to be discussed. Only practice can decide the matter, and tests of the relative performance of the NOT under the two methods were conducted as part of the experimentation procedure.

The NOT and Feasibility.--Problems exist for which the NOT is unable to reach the minimum solution, using permissible values for the w_i^0 and w_j^0, within a reasonable number of pivots. For some of these problems the NOT can be modified to find a solution which satisfies all the constraints, even though it may not give the theoretical minimum. To accomplish this it is necessary to relax the restrictions on the permissible values of the w_i^0 and w_j^0.

When the objective function reaches the cutoff level $B_0 > \langle b_0' \rangle + p_1$ which signals that p_0 was not safely chosen, instead of indicating a return to instruction (b) and a revision of the w^0 vector, the NOT procedure allows Gomory's algorithm to persist for a specified number of surplus pivots. Any solution found will be feasible, and can be recorded. Thereupon the NOT is set back on the course it would have followed if no digression had occurred.

So long as feasible solutions are permitted, it is expedient to go one step farther and allow the initial value of p_0 to be the negative quantity $b_0 - \langle b_0' \rangle$. This effectively makes the first values for the components of w^0, when w^0 is derived with respect to the final tableau, the same as the values of the variables in the simplex solution, rounded upward.

5. DESIGN OF EXPERIMENTS AND EMPIRICAL RESULTS.

The bulk of the problems on which the pivot rules and supplementary techniques were tested consisted of eight variables and eight inequalities. The objective function vector was randomly generated from the integers 1 through 29, and the absolute values of the coefficients of the constraint

variables randomly generated with range 0 to 29. Each entry in the A matrix and the b vector (referring to the notation of Formulation I, page 14) was made negative with probability .2. The absolute values of the c vector components varied randomly between 0 and 59. Problems which turned out to have no feasible solution have been omitted from the following discussion.

Because the random pivot rule was used with the supplementary techniques, and because ties occurring under other pivot rules for the choice of the pivotal column were broken by random selection, we have tested each rule* four times on each problem by altering the random number base. The first negative indicator rule, which has no random element, is of course excepted. An upper cutoff was applied uniformly to all the rules so that no problem was allowed to run for more than 400 pivots on any trial.

For purposes of comparing the performances of the rules, four considerations are relevant. Arranged in descending order of importance, they are: (1) the number of different problems that a rule solved at least once, (2) the total number of successful solution attempts, (3) the mean and standard deviation of the times required to get a solution, (4) the mean and standard deviation of the number of pivots required to reach a solution.

The rules were tested on sixty-eight problems, with decisive results. In spite of the simplicity of the problems no rule except the NOT was able to solve over half of them. The second best rule, the Largest Objective Function Change Rule, successfully solved only 32 of the 68 problems (47%) at least once. The NOT, by comparison, solved 52 problems (76%). By allowing feasible solutions not necessarily the minimum, the power of the NOT was extended further, enabling it to give answers (not necessarily optimal) to 63 (93%) of the problems.

*Henceforth the designation "rule" will refer both to the pivot rules and to the supplementary techniques.

In contrast to the wide difference between the NOT and the Largest Objective Function Change Rule, the remainder of the rules differ from each other by much smaller degrees, the ranking along the four scales becoming mixed. Interestingly, the rules were very nearly unanimous concerning the problems they failed to solve. Exactly half, or 34 of the 68 problems, were unsolved by any of the rules other than the NOT. The 24 problems solved by the Random Boost Rule, which was the worst, were also solved by all the others, and the 29 solved by the Ordinal Learning Rule included only two not among the 31 solved by the Maximum Frequency Eligible Rule. No other discrepancy was greater than this.

There were somewhat larger deviations among the totals of successful solution attempts, and reversals of ranking were not uncommon. While the Maximum Frequency Eligible Rule solved 2 problems more than the Ordinal Learning Rule, it had 8 fewer successful solution attempts.

Table I summarizes the information on the decision rules as they were applied to the 68 problems.

The data on means and standard deviations of times and pivots required to obtain solutions may sometimes appear to run counter to expectation. There are three reasons for this. First, for those rules that require evaluation of all eligible columns before selecting the column to be used for pivoting, the number of columns eligible in a tableau will make a difference in the time required to carry out a complete pivoting operation, thus causing the data for times to correspond inexactly to data on pivots. Second, times were recorded by the computer only to the nearest second. Means and deviations could be reduced or increased according to the direction in which various times were rounded. Third, for most of the rules, those problems which Gomory's algorithm was able to solve were solved with a mean of from 30 to 35 pivots.

Usually 60% or more of the successful solution attempts were accomplished in less than 30 pivots, and most of the rest in less than 60 pivots. However, about 3% to 5% of the successful solution attempts required 150 to 200 pivots. The occasional extreme numbers of pivots have produced standard deviations that seem unreasonably large. Since the mean times were generally 1/3 to 1/4 of the mean pivots, the effect of extreme times was not proportionately so great an influence on the standard deviations as the extreme number of pivots.

It is also possibly of interest to note that those problems which required over 150 pivots on one solution attempt were often solved in considerably fewer pivots on another attempt. As a rule, any problem solved was solved at least once in 30 pivots or less.

The data for the NOT is divided into three parts in Table I. The first part, designated "NOT Feasible," refers to all problems for which the NOT obtained a solution, whether or not the solution was optimal. The second part, designated simply "NOT," refers only to those problems for which the NOT did obtain the optimal solution (and gave notice of the fact). The third, called "NOT Restricted," refers to the New Origin Technique restricted to the 34 problems which were solved by at least one of the other rules. The data for the third part shows that the NOT found the problems solved by the other rules among the easier ones to handle.

It should be noted that the NOT was always used in the form that seeks feasible solutions. Hence the actual number of pivots allotted to solving for the theoretical minimum was somewhat less than 400. On the other hand, the figures for the pivots are based on the number of pivots required by the NOT to obtain the correct (or best) solution the first time, though the solution at that point was possibly only registered feasible.

Twenty, forty and eighty surplus pivots were tested for the NOT in hunting feasible solutions, with at the most a slight advantage to the eighty surplus pivots mark. One reason for this seems to be that the larger the

values for the new origin, the more quickly a feasible solution was reached, whether a good one or a bad one, so that earlier revision of the w_i^0 did not produce either defect or advantage for small values of p. Furthermore, Gomory's algorithm often seems to obey the principle whereby it either solves a problem in a fairly small number of pivots, or not at all. As already pointed out, those problems that did require a fairly large number of pivots on one solution attempt were usually solved in fewer pivots on the other attempts

The Random Restart Rule and the Random Boost Rule were similarly tested with upper cutoffs at 30, 50, and 70 pivots, with no significant difference in results. For example, using the 50 pivot cutoff, a problem solved on the first try in 45 pivots was generally matched, using the 30 pivot cutoff, by solving it on the second try within 10 to 20 pivots. The Random Restart Rule essentially multiplied the solution attempts of the Random Pivot Rule, adding further evidence that Gomory's algorithm by itself was not suited for the 34 problems unsolved by any of the rules except the NOT.

Additional Results for the NOT.--The form in which the NOT was programmed for the 68 problems referred to in Table I did not assign values to the slack variables. Thirty additional problems were tested with the NOT, using an upper cutoff of 200 pivots and four solution attempts for each problem, two by assigning values to the slacks and two not. For these problems, it appears that assigning values to the slacks which are close to their correct ones does not help the algorithm to reach a solution any more rapidly. This implies that the algorithm would have comparable difficulties solving a problem for which the simplex gave an integer answer.

On four of the 30 additional problems the use of slack assignments prevented the algorithm from reaching feasible solutions as good as those reached by the NOT in the form in which it ignored the slacks. On the other hand, the NOT without slack assignments was always at least as good, both in terms of

solutions obtained and in pivots required to reach them, as the NOT which gives values to the slacks, though generally there was no notable difference between the two.

In addition to the study of slack assignments, an examination was made of the comparative merits of the initial and final tableaus for producing new origins. Records kept of the new origin values obtained from the initial and final tableaus with the NOT indicate that the latter were generally far superior, as expected. Because of this double computation of w^0, however, the times recorded for the NOT are slightly distorted. It should particularly be noted that the times for problems solved by the NOT in only a few pivots may be misleading since the evaluations made before the start of Gomory's algorithm are the most extensive.

A record of the lower bounds established with the NOT did not always prove helpful in determining how large the deviation from the actual minimum, if it existed might be. In two cases the difference between a feasible solution and the lower bound was reduced to less than five, yet there was no indication that the minimum had been obtained. At the opposite extreme, eight of the problems which were solved correctly differed by as much as 40% from the lower bound at the point where the correct solution was first signalled feasible.

Table II has been designed to show the workings of the NOT in more detail. The four problems referred to in the table are not representative, but were chose to illustrate some of the variation in the results obtained by the NOT. The rows under each problem heading show the pivots, the time, the values of the w_i^0, the solution values of the variables, and the objective function value for each feasible solution reached in succession on a given trial. The new origins which have no other information recorded with them were not used successfully to reach a solution.

One trial only is illustrated for problems 1 and 2, while two trials are shown for problems 3 and 4. Problem 2 is the only one of the four for which the NOT obtained the minimum solution and recorded the fact. The most we know of the solutions for the other problems is that they are feasible.

The pivots for the NOT shown in Table II, unlike those for the NOT in Tables I and II, do not include the pivots required initially by the simplex method. Thus in Problem 1 the new origin that was first obtained gave a feasible solution without requiring any pivots with Gomory's algorithm. Problem 1 is interesting on two additional counts. No solution obtained after the first one was able to improve on it. Moreover, the algorithm devoted 380 pivots* to the fourth new origin, and obtained nothing, though the difference between this origin and the previous new origin was simply a unit reduction in the value of w_1^0.

The first new origin on Problem 2 did not give a feasible solution, and it took 8 surplus pivots to find one. The second origin was not only feasible, but turned out to be the minimum solution, though this was not verified until the answer had been gotten twice more, 45 pivots later.

Problem 3 demonstrates two more of the eccentricities of the NOT. Here the point of interest is in the difference between the results of the two solution trials. The best answer obtained in both trials was the same, and each of the w_i^* for this solution was greater than or equal to the w_i^0 of the new origin which produced it. The first solution obtained on the second trial, however, was better than the corresponding one of the first trial, though both were based on the same origin. The superior solution was obtained in fewer pivots and has the novel property that one of the variables which was given a nonzero value in the new origin was zero in the final answer.

---

* The simplex method required 4 pivots on this problem.

In Problem 4 the second trial agrees with the first on three of the feasible solutions, but registers one additional solution, which turns out to be the best one found. This solution, like the first one obtained in the second trial of Problem 3, was based on a new origin that could not safely have been relied upon to produce it. Nonetheless, none of the theoretically safe origins were successful in producing the solution on either trial. Such results suggest that a study of pivot sequencing for the NOT might prove more fruitful than for Gomory's algorithm alone.

Finally, a word about the NOT applied to problems of a larger size. We have received five problems from Ralph Gomory, and have tested the NOT and the Random Rule on four of them in the same way as the rules were tested on the eight variable eight inequality problems. (We currently do not have enough computer memory space to handle the fifth problem.) The Random Rule was unsuccessful in solving any of the four problems using the 400 pivot cutoff and making four solution attempts at each problem. The NOT obtained feasible solutions to three, though we don't know whether these solutions are optimal.

The results of the NOT on these problems are itemized in Table III. We suggest, however, that the NOT's performance on large problems in general may be somewhat different than for the problems used here. The reason is that the solutions which the NOT found were obtained by rounding the variables from the simplex tableaus upward. The structure of the problems makes it appear that the feasible solutions are good, but for an adequate test of the NOT, problems with optimal solutions substantially removed from any obtained by upward rounding should be tried.

TABLE I

(68 eight variable eight inequality problems)

Rule	(1) Problems Solved At Least Once (Total of 68 problems		(2) Successful Solution Trees (4 tries per problem, or 272 total)		(3) Times		(4) Pivots		(5) Approximate Rankings of the Rules by the 4 Criteria			
	No.	%	No.	% of tries on problems in (1)	Mean	SD	Mean	SD	(1)	(2)	(3)	(4)
1. NOT Feasible*	63	92.6	257	94.9	10.38	15.01	23.11	36.77				
1A. NOT	52	76.5	198	95.2	9.23	13.43	20.32	35.01	1	1	1	1
1B. NOT Restarted	44		111	99.3	4.80	6.62	10.3	14.16				
2. Max Objective Function Change	32	47.1	130	85.9	5.4	6.92	6.75	22.62	2	2	2	2
3. Random Restart**	31	45.6	106	85.5	6.34	7.67	26.45	41.50	3	3	3	3
4. Cardinal Learning	31	45.6	102	82.3	7.92	12.57	34.1	43.8	3	4	9	8
5. Random Rule	31	45.6	101	81.5	6.88	11.92	29.96	49.02	3	5	7	5
6. Max Frequency Eligible	31	45.6	93	75.0	6.[illegible]	7.[illegible]	32.62	3[illegible]	3	6	3	7
7. Ordinal Learning	29	42.6	[illegible]	87.1	8.26	14.00	34.57	[illegible]	[illegible]	5	[illegible]	9
8. First Negative Index *0**	26	38.2			5.55	8.32	16.34	49.28	[illegible]		8	[illegible]
9. NOT	25	36.8	85	85.0	6.65	7.62	30.80	33.58	[illegible]	8	5	6
10. Random Boost Rule**	24	35.3	91	94.8	6.[illegible]	8.03	28.34	33.62	[illegible]		6	[illegible]

* Eighty surplus pivots was allowed the NOT in each attempt to find a feasible solution which was possibly not optimal. Any surplus pivots used, however, reduced by the same amount the total number of pivots allowed before the NOT procedure was reached.

** The upper bound for pivots with the Random Restart Rule and Random Boost Rule was set at 50 pivots on each substitution with the 400 pivot principal cutoff retained as a means of multiplying the number of solution attempts.

*** The Worst Boost Index or Rule has no entries under heading (2) since

TABLE II*

Problems Illustrating Variations in the Behavior of the NOT

Problems	Pivots	Time (seconds)	New Origin Values of Variables								Feasible Solution Values of Variables								Objective Function Value
			0 W 1	0 W 2	0 W 3	0 W 4	0 W 5	0 W 6	0 W 7	0 W 8	W 1	W 2	W 3	W 4	W 5	W 6	W 7	W 8	
Problem 1	0	*	4	2	0	2	0	0	0	3	4	2	0	0	0	0	0	3	93
	2	6	3	2	0	0	0	0	0	2	4	2	0	0	0	0	0	3	98
	16	1	3	1	0	0	0	0	0	2	4	2	0	0	0	0	0	3	98
	*		2	1	0	0	0	0	0	2									
Problem 2	8	5	1	0	2	5	0	0	0	0	1	0	3	5	0	0	0	0	45
	8	7	1	0	2	4	0	0	0	0	1	0	2	4	0	0	0	0	35
	19	1	0	0	2	3	0	0	0	0	1	0	2	4	0	0	0	0	25
	53	2	0	0	1	2	0	0	0	0	1	0	2	4	0	0	0	0	35
Problem 3	19	5	0	0	2	1	3	1	0	0	1	0	4	1	4	1	0	0	17
Trial 1	32	8	0	0	1	1	3	0	0	0	0	0	1	1	4	0	1	0	12
	*		0	0	1	0	3	0	0	0									
Trial 2	12	4	0	0	2	1	3	1	0	0	1	0	3	1	4	0	0	0	15
	26	8	0	0	1	1	3	0	0	0	0	0	1	1	4	0	1	0	7
	*		0	0	1	0	3	0	0	0									
	*		0	0	0	1	3	0	0	0									
Problem 4	7	6	0	5	0	2	3	0	0	4	0	5	0	6	3	0	0	6	33
Trial 1	11	7	0	4	0	2	3	0	0	4	0	4	0	5	3	0	0	5	31
	16	9	0	3	0	2	3	0	0	4	0	3	0	5	3	0	0	5	41
	*		0	2	0	2	3	0	0	4									
	*		0	0	0	2	2	0	0	4									
Trial 2	7	3	0	5	0	2	3	0	0	4	0	5	0	6	3	0	0	6	33
	11	4	0	4	0	2	3	0	0	4	0	4	0	5	3	0	0	5	31
	16	6	0	3	0	2	3	0	0	4	0	3	0	5	3	0	0	5	41
	25	9	0	2	0	2	3	0	0	4	0	0	1	6	4	0	0	4	29
	*		0	0	0	2	2	0	0	4									

* The New Origin associated with this row did not produce a feasible solution within the pivot cutoff limit.

TABLE III

The NOT Applied to Gomory's Problems

Problem	Problem Size		Objective Function	
	Variables	Inequalities	Simplex Solution	NOT Feasible Solution
I	15	5	5,075	29
II	25	40	*	15
III	30	31	5,833	13
IV	50	12	3,958,660	*

* The NOT obtained no feasible solution, but gave 20.34 as a lower bound on the optimal integer solution

APPENDIX A

The All Integer Integer Programming Algorithm.

Form of the Problem. Minimize $wb + b_0$ subject to $wA \geq c$, $w \geq 0$, w integer

In the above notation w and b are row vectors, b and c are column vectors, b_0 is a scalar, and A is a matrix, all of which are entirely integer, and of dimensions such that the problem is well-defined.

Initial Tableau

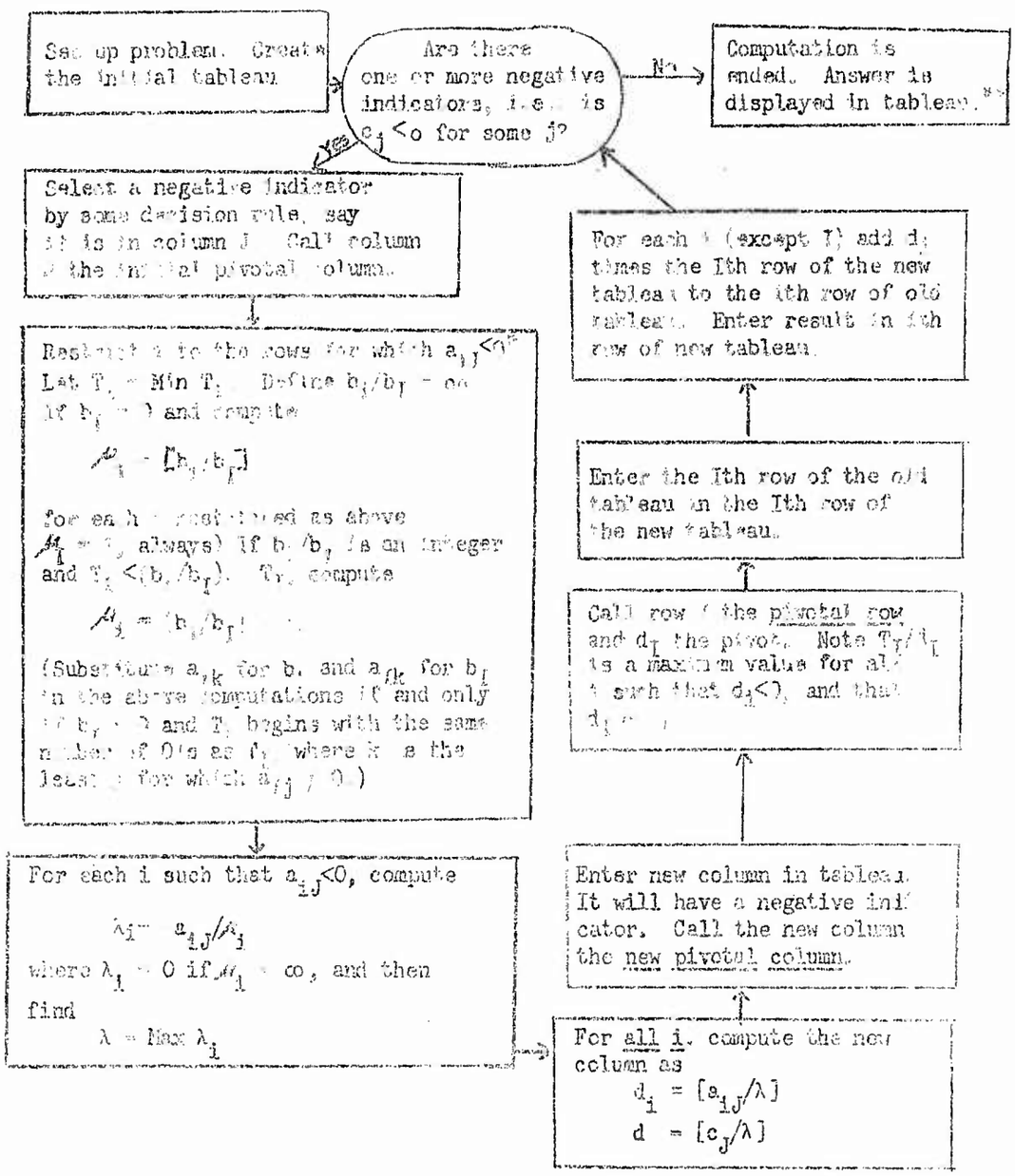
b	A	- I
$-b_0$	c	0

Let T denote the "tableau matrix" formed by annexing the column vector b to the left of A and -I, as in the upper portion of the tableau above. For any subsequent tableau T will continue to denote the matrix found in the same location. Let T_i represent the vector corresponding to the i th row of T. By $T_i \leq T_j$ we will mean that T_i is lexicographically less than T_j.

Gomory's algorithm assumes that $T_i > 0$ (lexicographically) for all i (if they are not, a method exists for making them so). The algorithm is called the all integer method because all coefficients in the tableau remain integer during the entire calculation given that the entries in the initial tableau were integers.

For simplicity in describing Gomory's algorithm in the diagram which follows, let the matrix initially consisting of A and -I be denoted simply by A, and let the vector initially consisting of c and 0 be denoted by c. Again, in subsequent tableaus, A and c will denote the matrix and vector found in the same locations as the original A and c.

DIAGRAM FOR GOMORY'S ALGORITHM



* If $a_{iJ} \geq 0$ for all i (when $c_J < 0$), the problem has no solution.

** The objective function value (negatively signed) appears in the location initially occupied by b_o, and the solution vector $w = w^*$ appears in the partition of the bottom row initially occupied by the c_j's.

APPENDIX B

Theorem 4. Let $\Sigma w_k a_k = c$ represent the constraint chosen for pivot from a simplex tableau T. Let the "prime" superscript (') denote values pertaining to the tableau T' derived from T by the pivot operation; for example, $\Sigma w_k a_k' = c'$ refers to the pivotal constraint from T as it appears in T'. Finally, let w_i^0 and w_j^0 denote the components of w^0 derived from the pivotal constraint in T, and let $(w_{i'}^0)'$ and $(w_{j'}^0)'$ denote the components of w^0 obtained from the same constraint as it appears in T'. Then $i = j'$, $j = i'$, and $w_i^0 = (w_{j'}^0)'$, $w_j^0 = (w_{i'}^0)'$.

Proof: To show this it is necessary first to show that if $a_m/b_m \leq a_n/b_n$ for the pivotal constraint, then $-a_{m'}/b_m \leq -a_{n'}/b_{n'}$ after the pivot. This implies in particular that $j' = i$. The simplex algorithm defines $a_{k'} = a_k/a_i$ and $b_{k'} = b_k - b_i a_k/a_i$, where $a_j/b_i = \min(a_k/b_k)$ for $a_k<0$. The subscript i, it should be noted, is defined the same for the pivot element a_i and the new origin value w_i^0 of Theorem 3 for any constraint which is eligible for pivoting. We then may write

$$(7) \quad -a_{k'}/b_{k'} = (-a_k/a_i)/(b_k - b_i a_k/a_i) = 1/b_i + (a_i/b_i)/((a_k/b_k)b_i - a_i).$$

Inspection of this latter form shows the assertion of order preservation to be valid. Thus $j' = i$.

We now observe that

$$\Omega' = -a_{r'}/b_{r'} \text{ where } -a_{r'}/b_{r'} = \min(-a_{k'}/b_{k'}),\ k \neq i.$$

Rearranging (7) once again, Ω' may be written as

$$\Omega' = (a_r/b_r)/((a_r/b_r)b_i - a_i) = 1/(b_i - a_i)$$

with r substituted for k. We use the following relations from the simplex:

$c' = c/a_i;\ a_{i,} = 1;\ B' = B - cb_i/a_i;\ b_{i,} = 0.$

Then

$$(\Omega B' + c')/(\Omega' b_{i,} + a_{i,}) = HB'/(Hb_i - a_i) = \frac{(c/a_i)(\quad - Hb_i)}{(Hb_i - a_i)} =$$

$$(HB - c)/(Hb_i - a_i)$$

from which it follows that $(w^0_j)' = w^0_i$. The identity $(w^0_{i,})' = w^0_j$ is obtained in a like fashion.

REFERENCES

1. Kuhn, Harold W. and Quandt, Richard E. "An Experimental Study of the Simplex Method," mimeographed notes, Princeton University, 1962.

2. Gomory, Ralph E. "All Integer Integer Programming Algorithm," IBM Research Report RC-189, International Business Machines Corporation Research Center, Yorktown Heights, 1960

3. Bush, R. R., Mosteller, Frederick, and Thompson, G. I "A Formal Structure for Multiple-Choice Situations," Decision Processes, Thrall, Coombs and Davis, Eds., John Wiley and Sons, Inc., New York, 1954.

4. Fischer, Henry and Thompson, Gerald. "Probabilistic Learning Combinations of Local Job Shop Scheduling Rules," O.N.R. Research Memorandum No. 60, Carnegie Institute of Technology, 1961.